



Improving program performance with AMD CodeAnalyst for Linux®

Paul J. Drongowski
AMD CodeAnalyst team

May 9, 2007

AMD CodeAnalyst for Linux®

- Identify and investigate performance hot-spots
- Easy to use graphical user interface
 - Simplified configuration of performance events
 - GUI is consistent for cross-platform use
- Low overhead, system-wide data collection
 - Can profile dynamically loaded modules
 - Can profile kernel mode drivers as well as applications
- Results are displayed in configurable views
 - Source and disassembly annotated with performance data
 - Rates and ratios assist interpretation of results

AMD CodeAnalyst for Linux

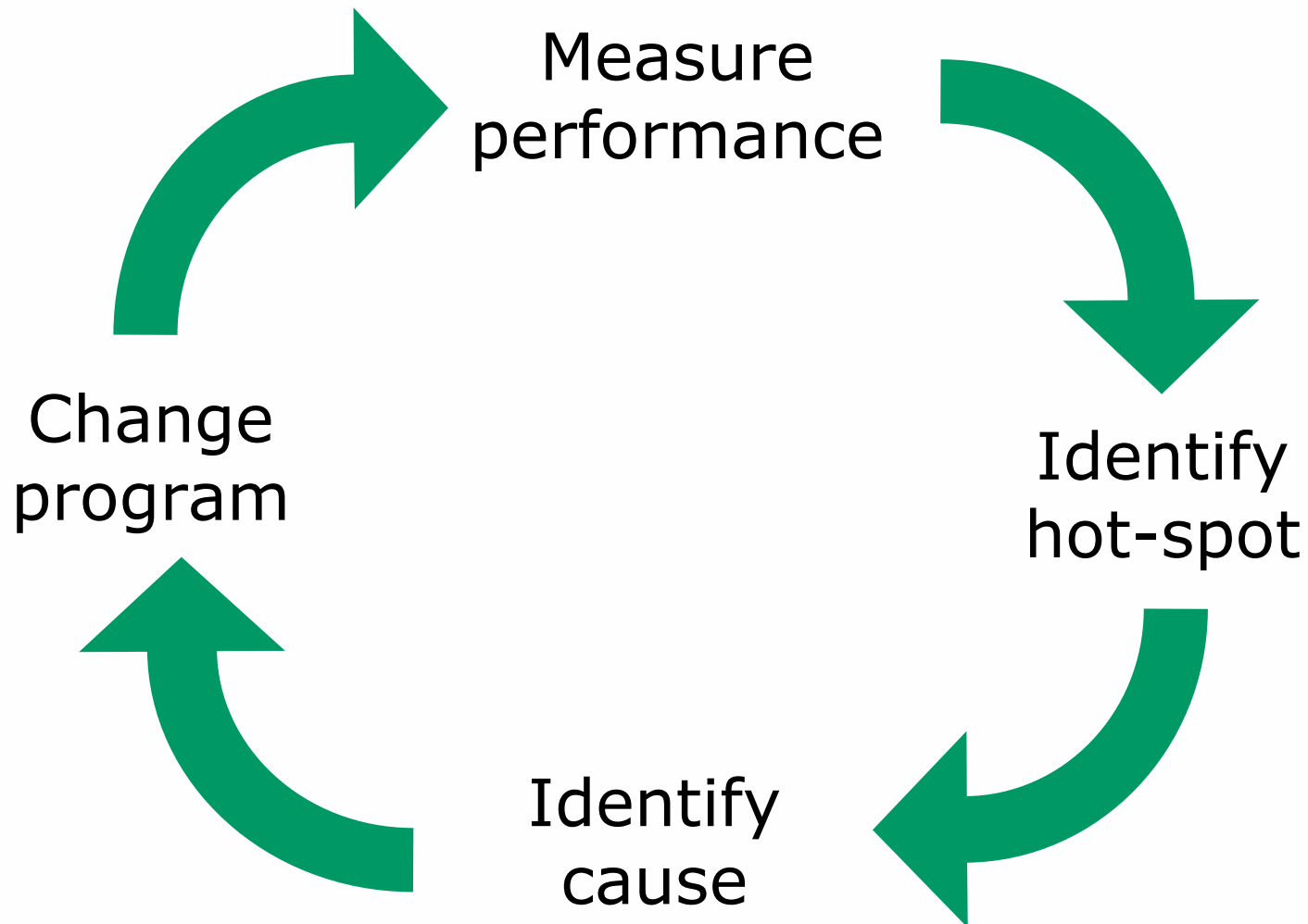
- Open source
 - Uses oprofile for data collection
 - Can import oprofile data
 - CodeAnalyst GUI is Qt-based
- Java profiling support
 - Version 2.5: Modified oprofile driver and daemon
 - Version 2.6: Modified daemon only

Presentation outline

- AMD CodeAnalyst in action
 - Time-based profiling
 - Event-based profiling
 - Tuning
- Downloading AMD CodeAnalyst
- On-line resources

AMD CodeAnalyst in action

Performance tuning cycle



Kinds of analysis

- Identify hot-spot: **Time-based profiling**
 - Where is the application or system spending time?
- Identify cause: **Event-based profiling**
 - How is the application behaving on the CPU? Memory?
 - Are there performance issues in a code region?

Example program

- Classic example: Textbook matrix multiply
- Long memory strides cause data cache and TLB misses

```
void multiply_matrices()
{
    // Multiply the two matrices
    for (int i = 0 ; i < N ; i++) {
        for (int j = 0 ; j < M ; j++) {
            float sum = 0.0 ;
            for (int k = 0 ; k < K ; k++) {
                sum = sum + matrix_a[i][k] * matrix_b[k][j] ;
            }
            matrix_r[i][j] = sum ;
        }
    }
}
```

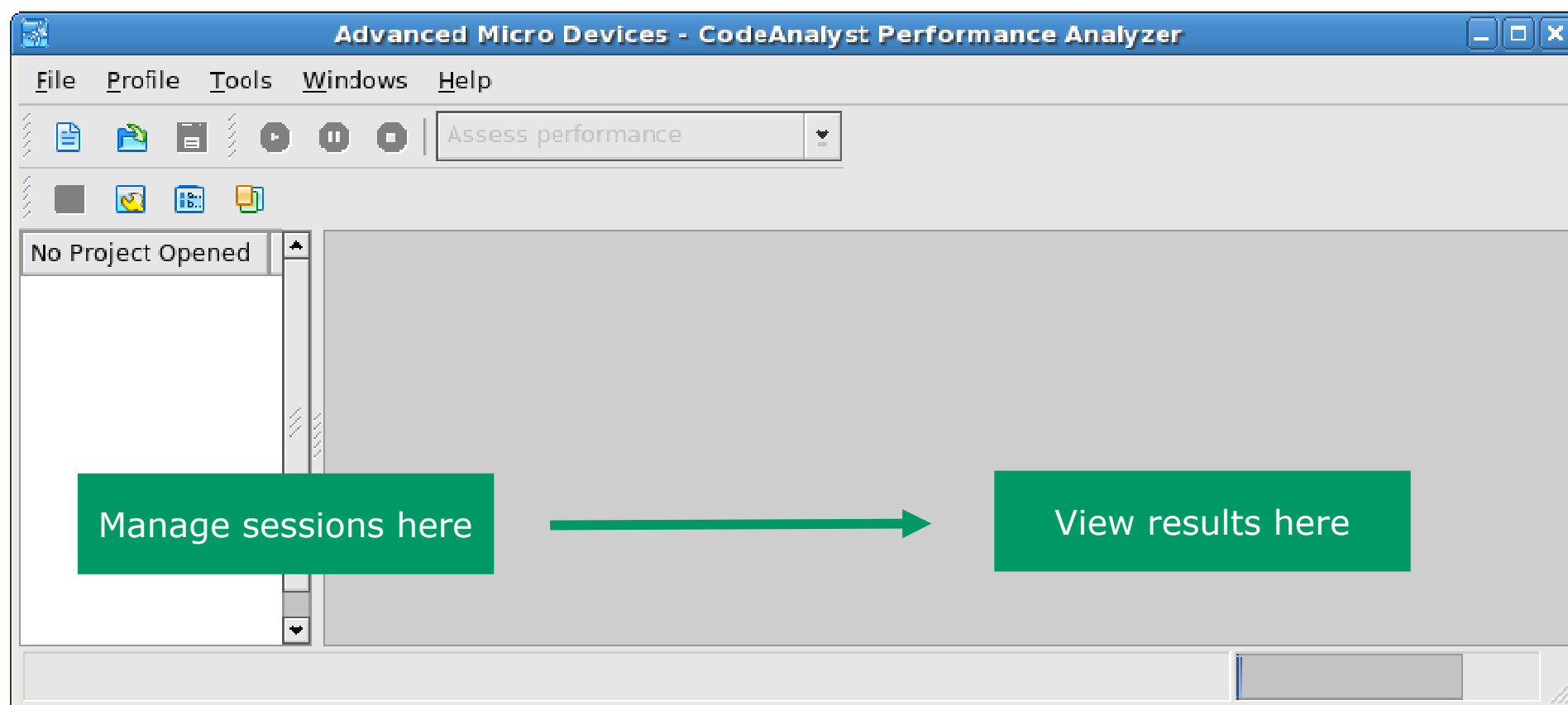

Program preparation

- AMD CodeAnalyst samples all running software components
 - Applications, libraries, kernel modules, ...
- No explicit preparation is necessary
- However, symbolic information is needed to breakdown samples by function or source line
- Compile and build application with debug information

```
g++ -O2 -g -o classic classic.cpp
```

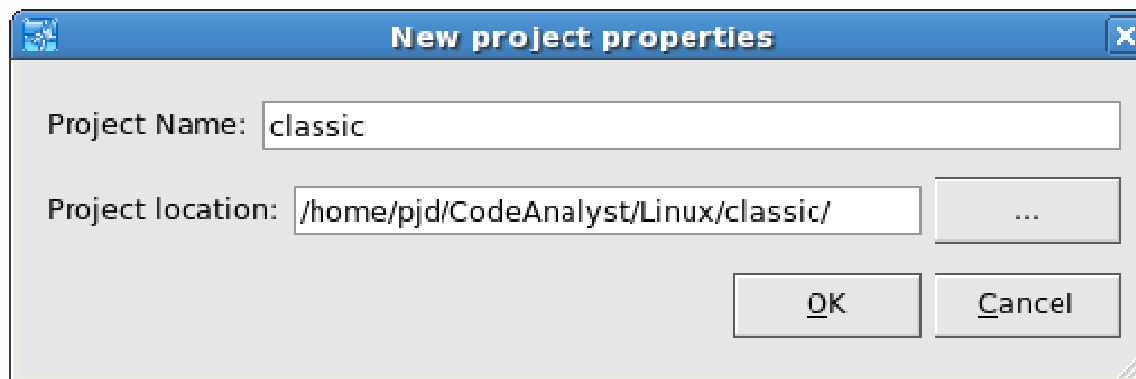
Graphical user interface

- Uses a project- and session-oriented user model
- First step: Create a new project



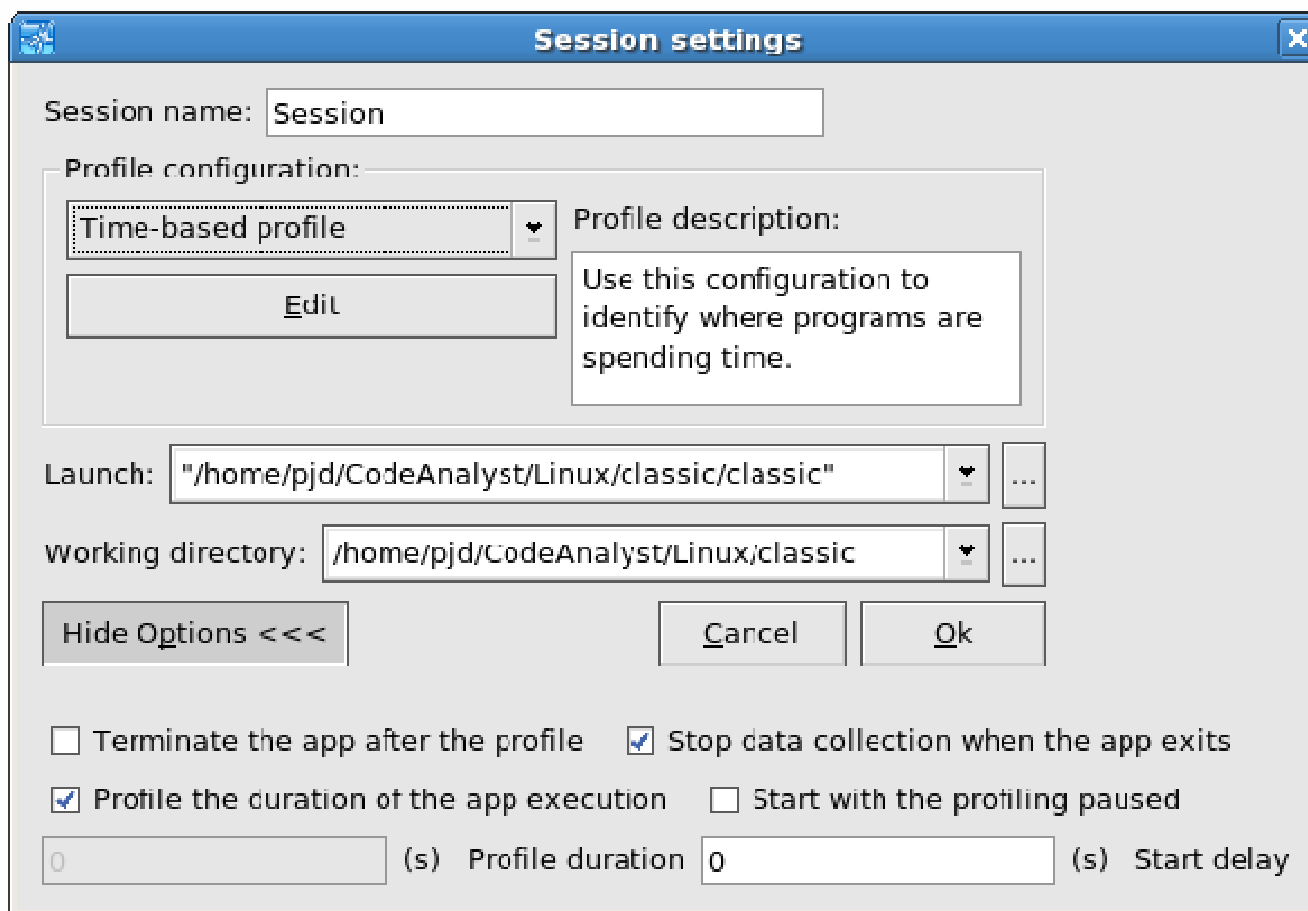
Enter project information

- Choose a name for the new project
- Choose a directory to hold the project



Prepare to collect data

- Enter the path to the application program to analyze and set the working directory
- Choose the kind of analysis to perform

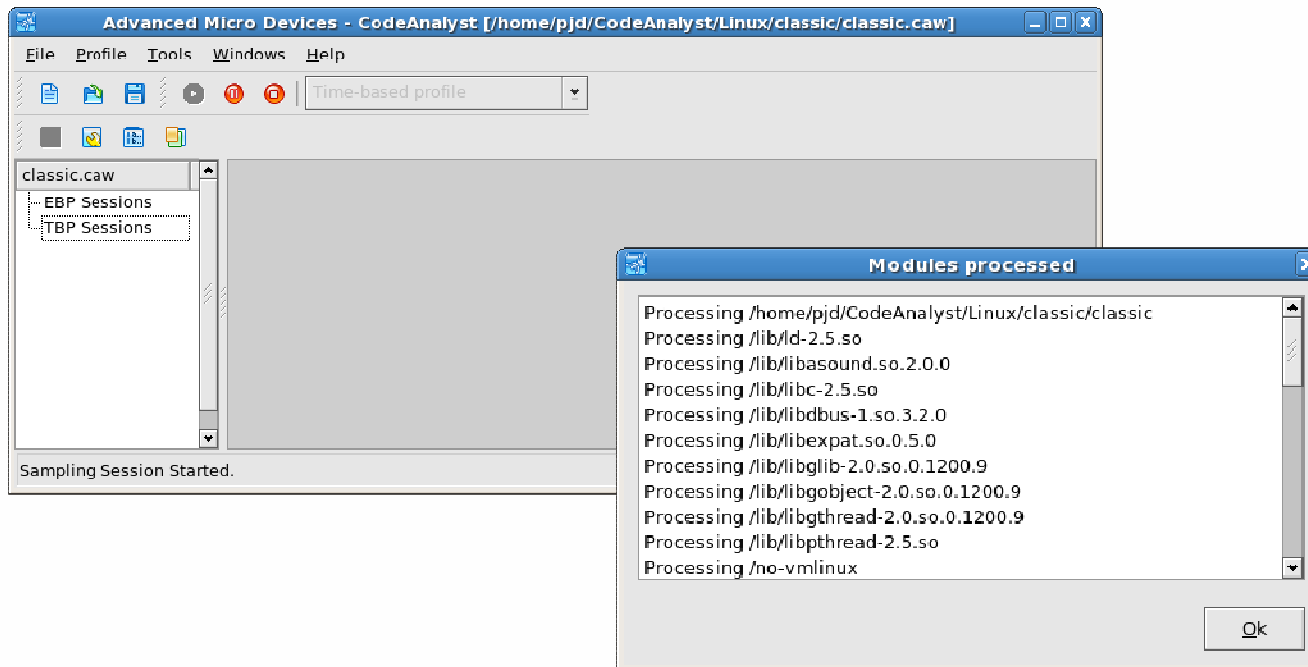


The image shows a 'Session settings' dialog box with the following fields and options:

- Session name:** Session
- Profile configuration:**
 - Time-based profile (selected)
 - Edit button
- Profile description:** Use this configuration to identify where programs are spending time.
- Launch:** "/home/pjd/CodeAnalyst/Linux/classic/classic" (with a browse button)
- Working directory:** /home/pjd/CodeAnalyst/Linux/classic (with a browse button)
- Buttons:** Hide Options <<<, Cancel, Ok
- Checkboxes:**
 - ☐ Terminate the app after the profile
 - ☒ Stop data collection when the app exits
 - ☒ Profile the duration of the app execution
 - ☐ Start with the profiling paused
- Input fields:**
 - 0 (s) Profile duration
 - 0 (s) Start delay

Start data collection

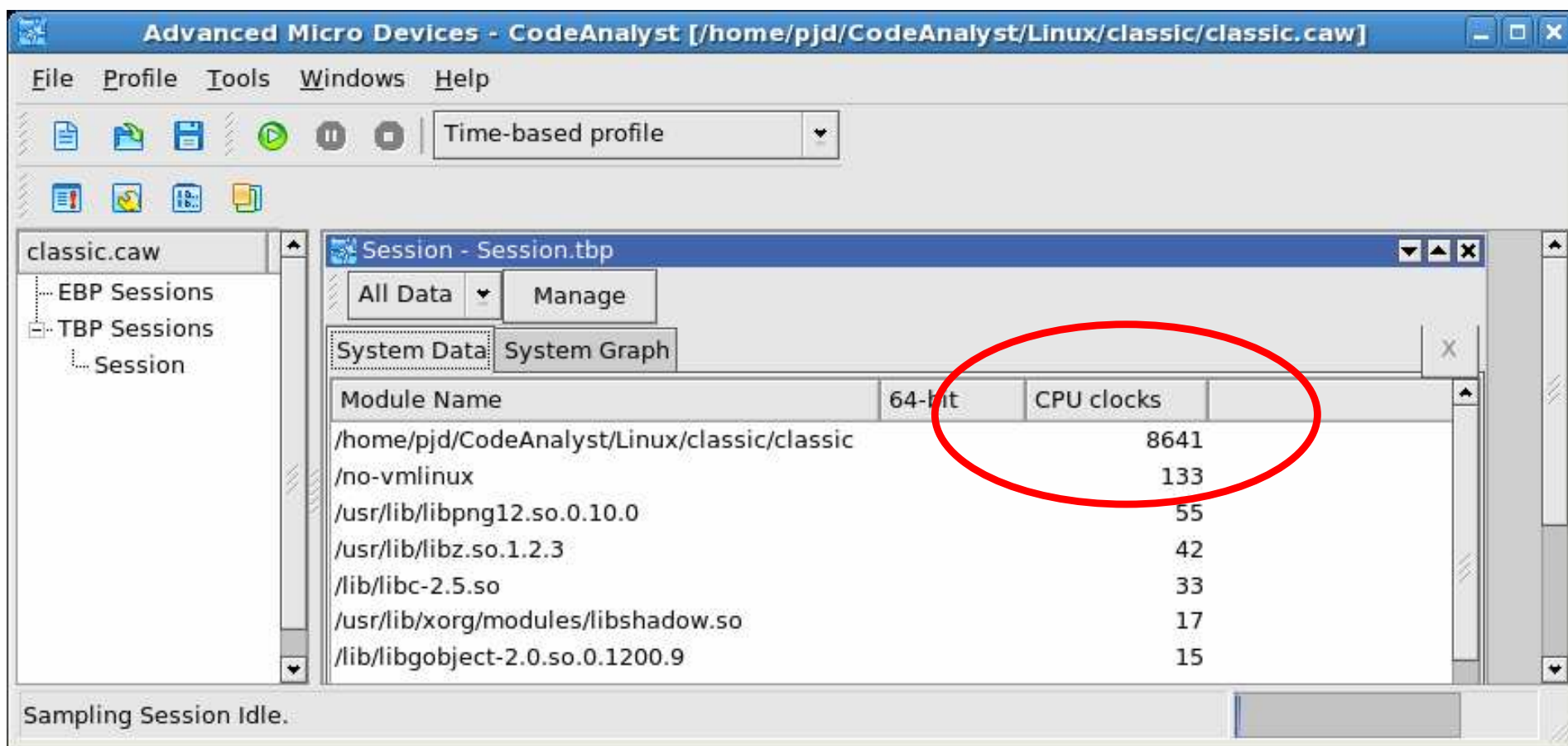
- Click the start button in the toolbar
- AMD CodeAnalyst runs the application and data is collected
- AMD CodeAnalyst imports the data and displays results



Time-based profiling

Time-based profiling

- Shows time for each module
- Time is measured by sampling PMC event 0x76 (CPU clocks)
- Double click module to drill down to its functions



Advanced Micro Devices - CodeAnalyst [/home/pjd/CodeAnalyst/Linux/classic/classic.caw]

File Profile Tools Windows Help

Time-based profile

classic.caw

- EBP Sessions
- TBP Sessions
 - Session

Session - Session.tbp

All Data Manage

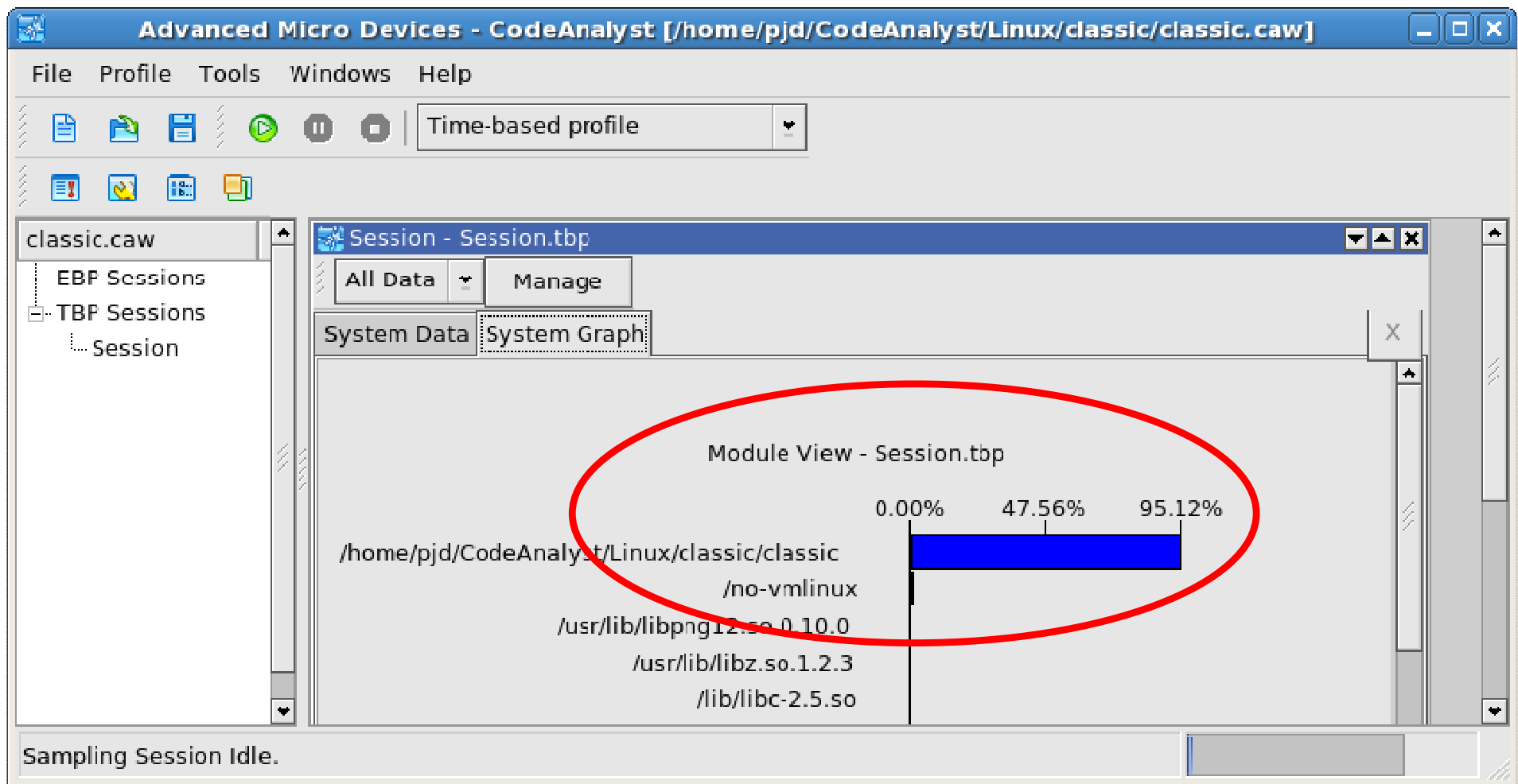
System Data System Graph

| Module Name | 64-bit | CPU clocks |
|---|--------|------------|
| /home/pjd/CodeAnalyst/Linux/classic/classic | | 8641 |
| /no-vmlinux | | 133 |
| /usr/lib/libpng12.so.0.10.0 | | 55 |
| /usr/lib/libz.so.1.2.3 | | 42 |
| /lib/libc-2.5.so | | 33 |
| /usr/lib/xorg/modules/libshadow.so | | 17 |
| /lib/libgobject-2.0.so.0.1200.9 | | 15 |

Sampling Session Idle.

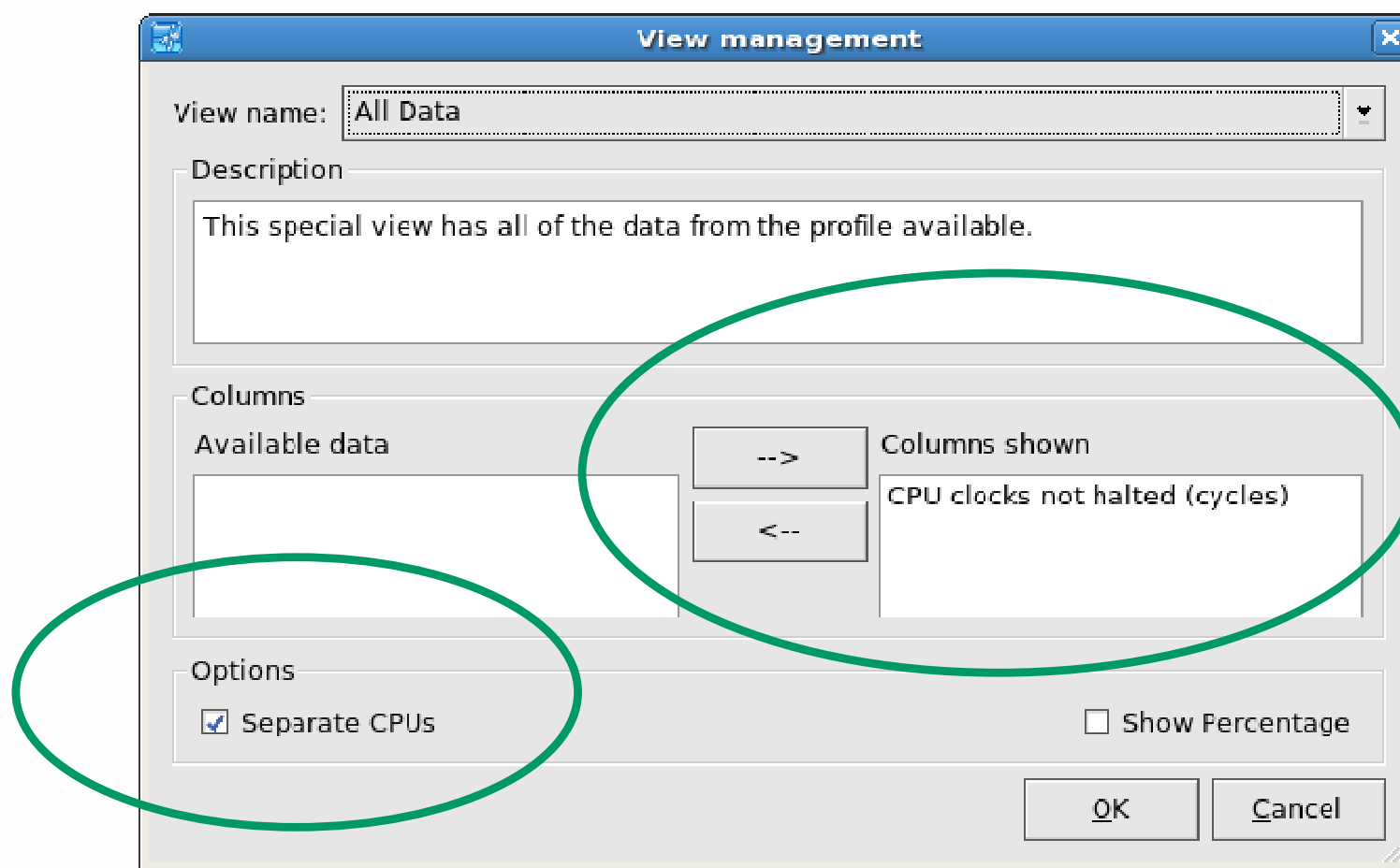
Time-based profiling

- Show results in bar chart form, too



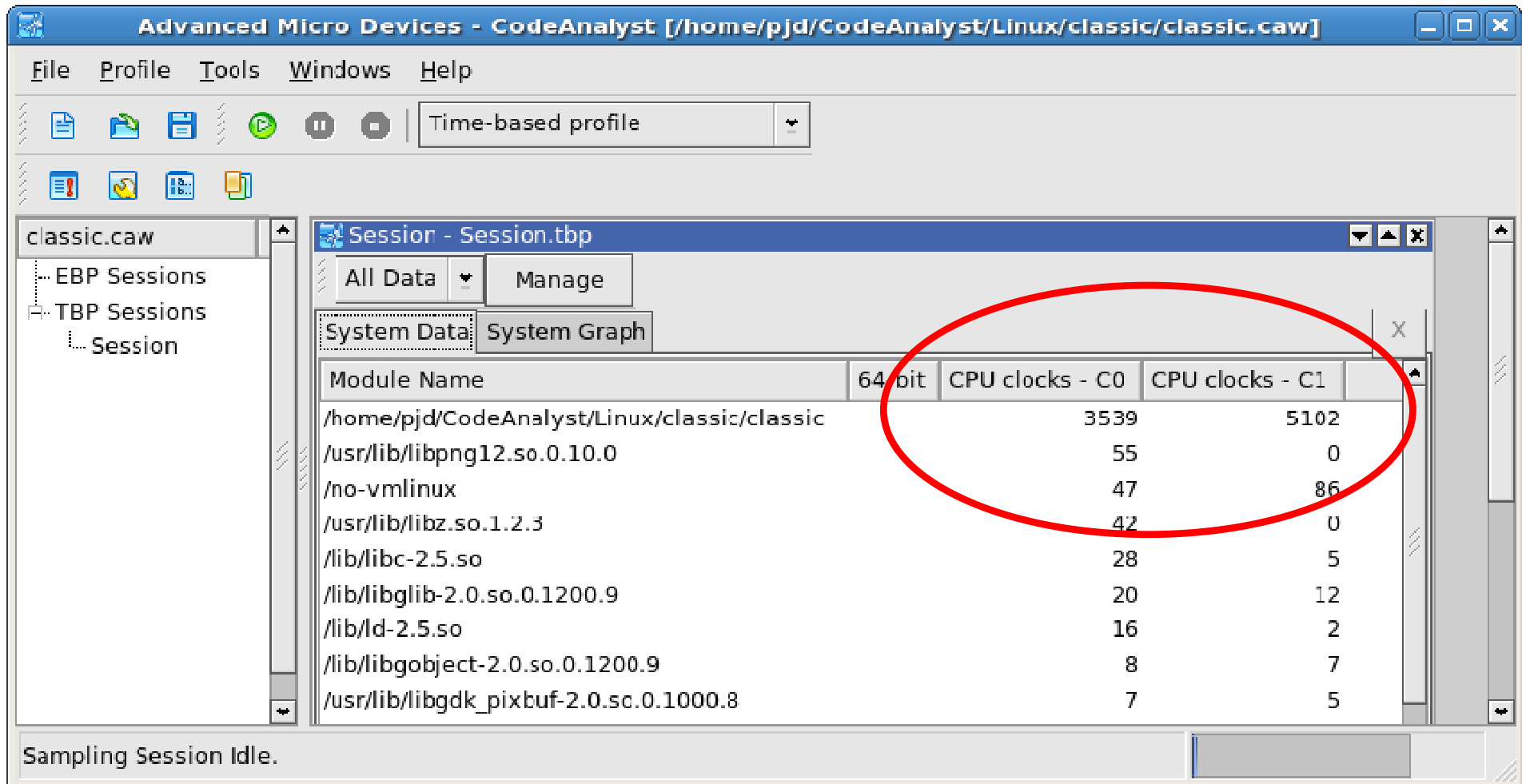
Use “Manage” button to adjust view

- Add or remove columns to be shown
- Aggregate data or separate data by CPU



TBP data shown for each CPU

- Shows time spent on each CPU
- Indicates degree of processor affinity during execution



Advanced Micro Devices - CodeAnalyst [/home/pjd/CodeAnalyst/Linux/classic/classic.caw]

File Profile Tools Windows Help

Time-based profile

classic.caw

- EBP Sessions
- TBP Sessions
 - Session

Session - Session.tbp

All Data Manage

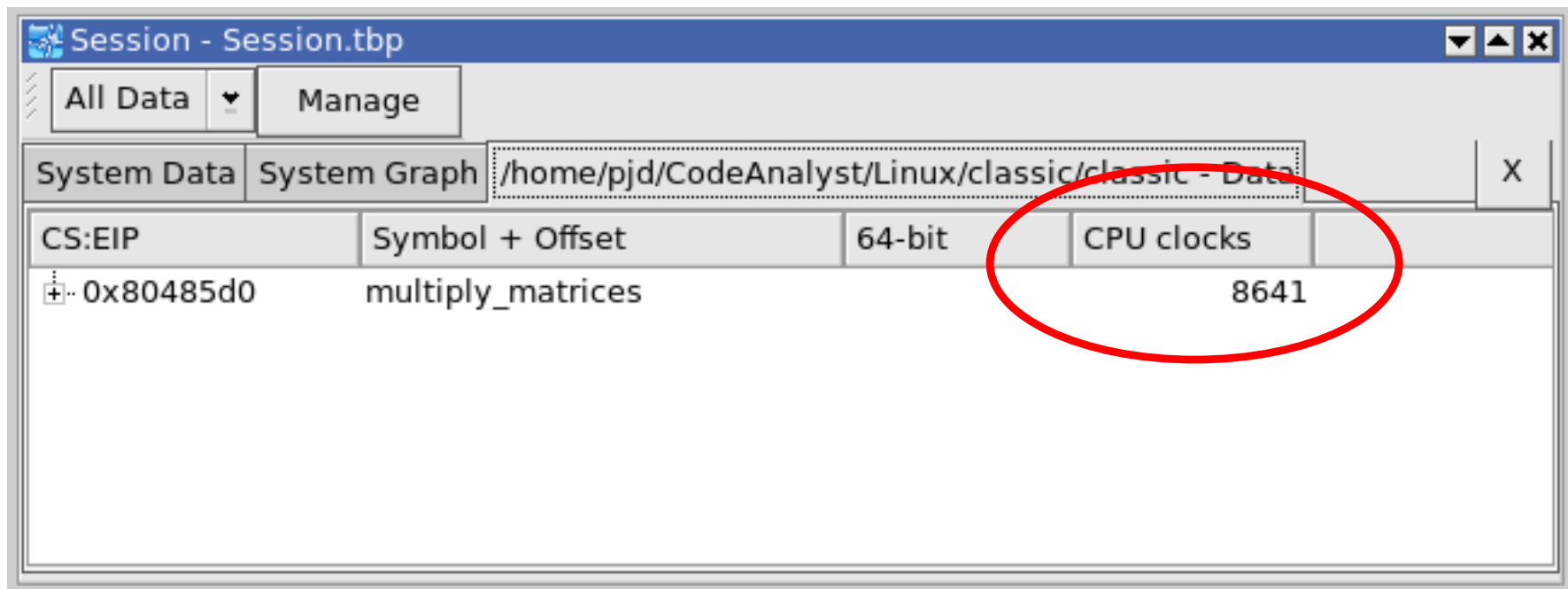
System Data System Graph

| Module Name | 64 bit | CPU clocks - C0 | CPU clocks - C1 |
|---|--------|-----------------|-----------------|
| /home/pjd/CodeAnalyst/Linux/classic/classic | | 3539 | 5102 |
| /usr/lib/libpng12.so.0.10.0 | | 55 | 0 |
| /no-vmlinux | | 47 | 86 |
| /usr/lib/libz.so.1.2.3 | | 42 | 0 |
| /lib/libc-2.5.so | | 28 | 5 |
| /lib/libglib-2.0.so.0.1200.9 | | 20 | 12 |
| /lib/ld-2.5.so | | 16 | 2 |
| /lib/libgobject-2.0.so.0.1200.9 | | 8 | 7 |
| /usr/lib/libgdk_pixbuf-2.0.so.0.1000.8 | | 7 | 5 |

Sampling Session Idle.

Time-based profiling

- Drill down to functions within a module
- Shows a function-by-function time breakdown
- Identify candidates for investigation and improvement



The screenshot shows the AMD CodeAnalyst interface with a window titled "Session - Session.tbp". The "All Data" dropdown is set to "Manage". The "System Data" tab is active, showing a table of profiling data. The table has columns for "CS:EIP", "Symbol + Offset", "64-bit", and "CPU clocks". The "CPU clocks" column is circled in red. The data row shows the address "0x80485d0" for the function "multiply_matrices", which took 8641 CPU clocks.

| CS:EIP | Symbol + Offset | 64-bit | CPU clocks |
|-----------|-------------------|--------|------------|
| 0x80485d0 | multiply_matrices | | 8641 |

Time-based profiling

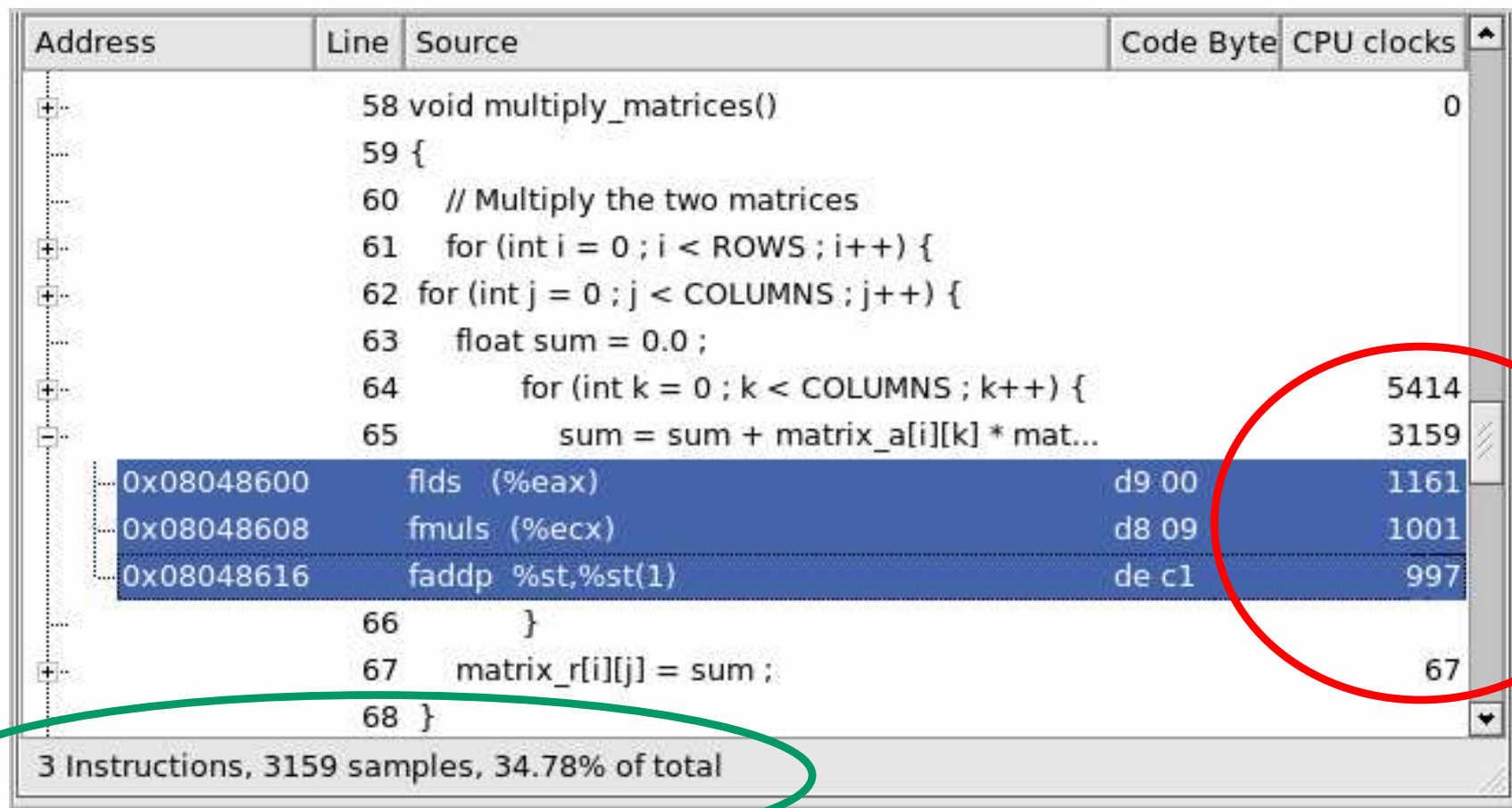
- Drill down to source code
- Identify code region to investigate and tune

| Address | Line | Source | Code Bytes | CPU clocks |
|---------|------|---------------------------------------|------------|------------|
| 58 | | void multiply_matrices() | | 0 |
| 59 | | { | | |
| 60 | | // Multiply the two matrices | | |
| 61 | | for (int i = 0 ; i < ROWS ; i++) { | | |
| 62 | | for (int j = 0 ; j < COLUMNS ; j++) { | | |
| 63 | | float sum = 0.0 ; | | |
| 64 | | for (int k = 0 ; k < COLUMNS ; k++) { | | 5414 |
| 65 | | sum = sum + matrix_a[i][k] * mat... | | 3159 |
| 66 | | } | | |
| 67 | | matrix_r[i][j] = sum ; | | 67 |
| 68 | | } | | |
| 69 | | } | | |
| 70 | | } | | 1 |
| 71 | | | | |

0 Instruction, 0 samples, 0.00% of total

Time-based profiling

- Drill down to instructions too
- Select instructions to see a summary in the info bar

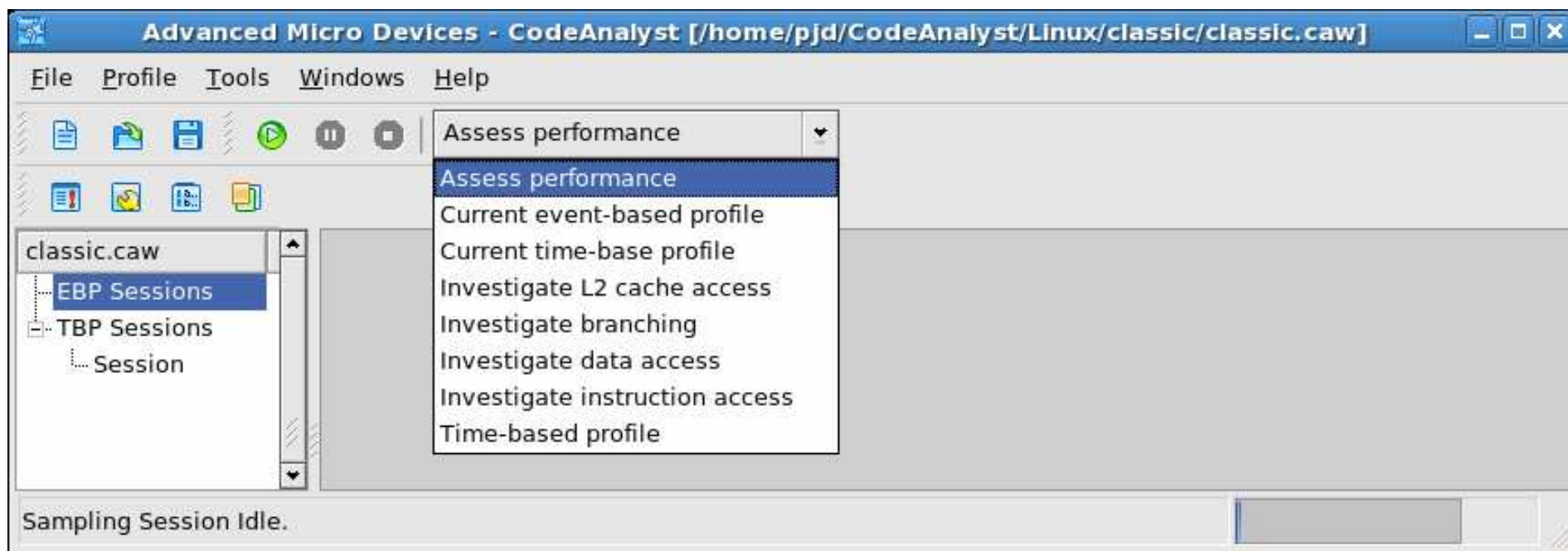


| Address | Line | Source | Code Byte | CPU clocks |
|---|------|---------------------------------------|-----------|------------|
| | 58 | void multiply_matrices() | | 0 |
| | 59 | { | | |
| | 60 | // Multiply the two matrices | | |
| | 61 | for (int i = 0 ; i < ROWS ; i++) { | | |
| | 62 | for (int j = 0 ; j < COLUMNS ; j++) { | | |
| | 63 | float sum = 0.0 ; | | |
| | 64 | for (int k = 0 ; k < COLUMNS ; k++) { | | 5414 |
| | 65 | sum = sum + matrix_a[i][k] * mat... | | 3159 |
| 0x08048600 | | flds (%eax) | d9 00 | 1161 |
| 0x08048608 | | fmuls (%ecx) | d8 09 | 1001 |
| 0x08048616 | | faddp %st,%st(1) | de c1 | 997 |
| | 66 | } | | |
| | 67 | matrix_r[i][j] = sum ; | | 67 |
| | 68 | } | | |
| 3 Instructions, 3159 samples, 34.78% of total | | | | |

Event-based profiling

Event-based profiling

- Observe how program behaves on CPU and memory
- Develop and test hypothesis about a performance issue
- Choose the type of analysis to perform
- Data display and drill down are similar to TBP



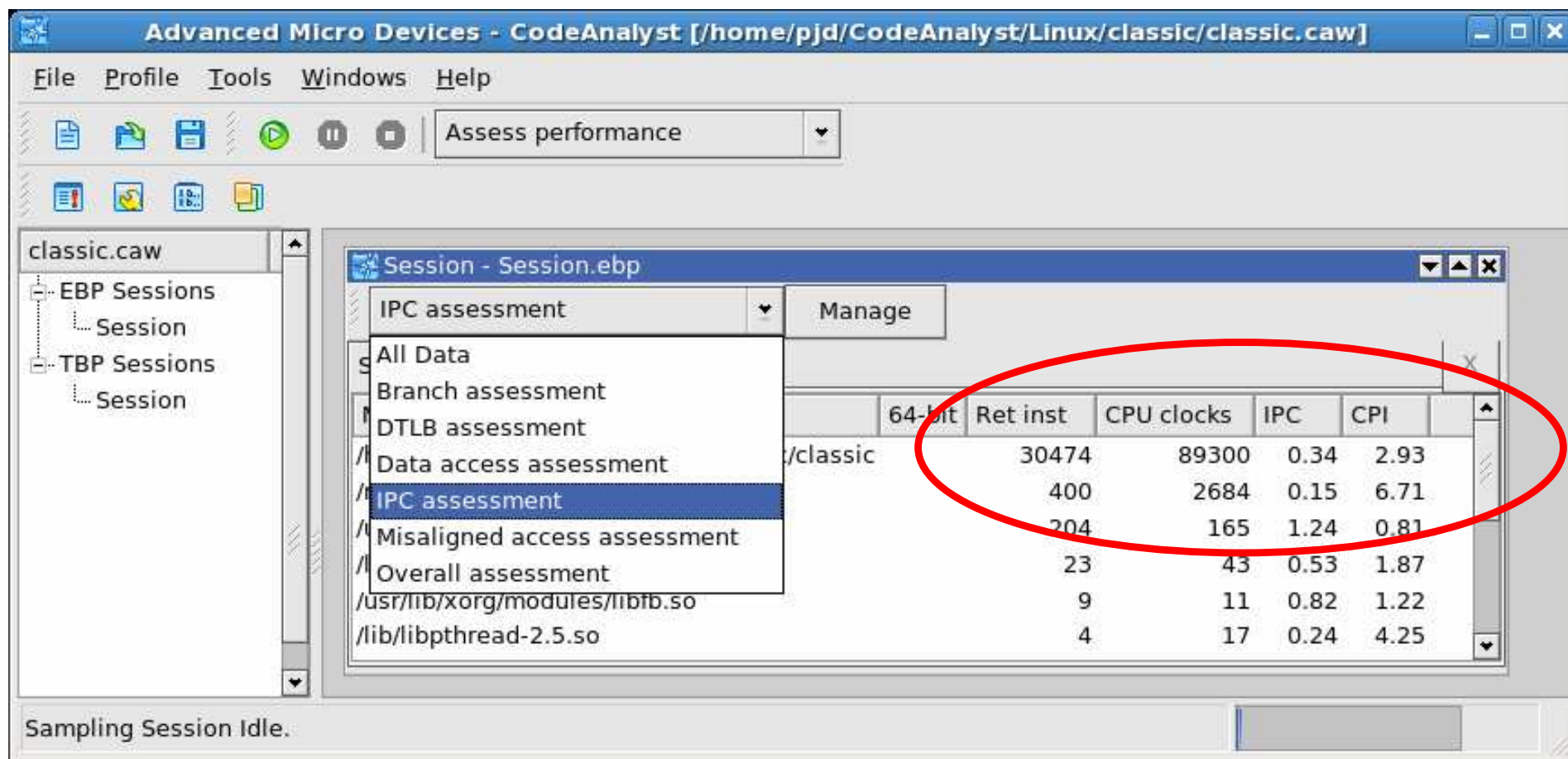
Kinds of analysis

- Choose from predefined profile configurations
- Define a customized configuration ("Current")

| Profile configuration | Focus |
|--------------------------------|-----------------------------------|
| Time-based profile | Hot-spot identification |
| Assess performance | Overall evaluation ("triage") |
| Investigate data access | Memory access patterns / locality |
| Investigate L2 cache access | Memory access patterns / locality |
| Investigate branching | Branch and return prediction |
| Investigate instruction access | Code placement / locality |

Event-based profiling

- Views are offered depending upon available event data
- Views show computed measurements such as ratios
- IPC view shows “Instructions per cycle”



The screenshot shows the AMD CodeAnalyst interface. The main window displays a table of performance metrics for various components. A red circle highlights the 'IPC assessment' row, which shows the following data:

| Component | 64-bit | Ret inst | CPU clocks | IPC | CPI |
|--------------------------------|--------|----------|------------|------|-----|
| /usr/lib/xorg/modules/libfb.so | 30474 | 89300 | 0.34 | 2.93 | |
| /lib/libpthread-2.5.so | 400 | 2684 | 0.15 | 6.71 | |
| /usr/lib/xorg/modules/libfb.so | 204 | 165 | 1.24 | 0.81 | |
| /lib/libpthread-2.5.so | 23 | 43 | 0.53 | 1.87 | |
| /usr/lib/xorg/modules/libfb.so | 9 | 11 | 0.82 | 1.22 | |
| /lib/libpthread-2.5.so | 4 | 17 | 0.24 | 4.25 | |

Event-based profiling

- DTLB view shows the DTLB miss rate and ratio
- DTLB miss rate is high – 1 miss per 10 instructions
- Low IPC is due to DTLB misses

Session - Session.ebp

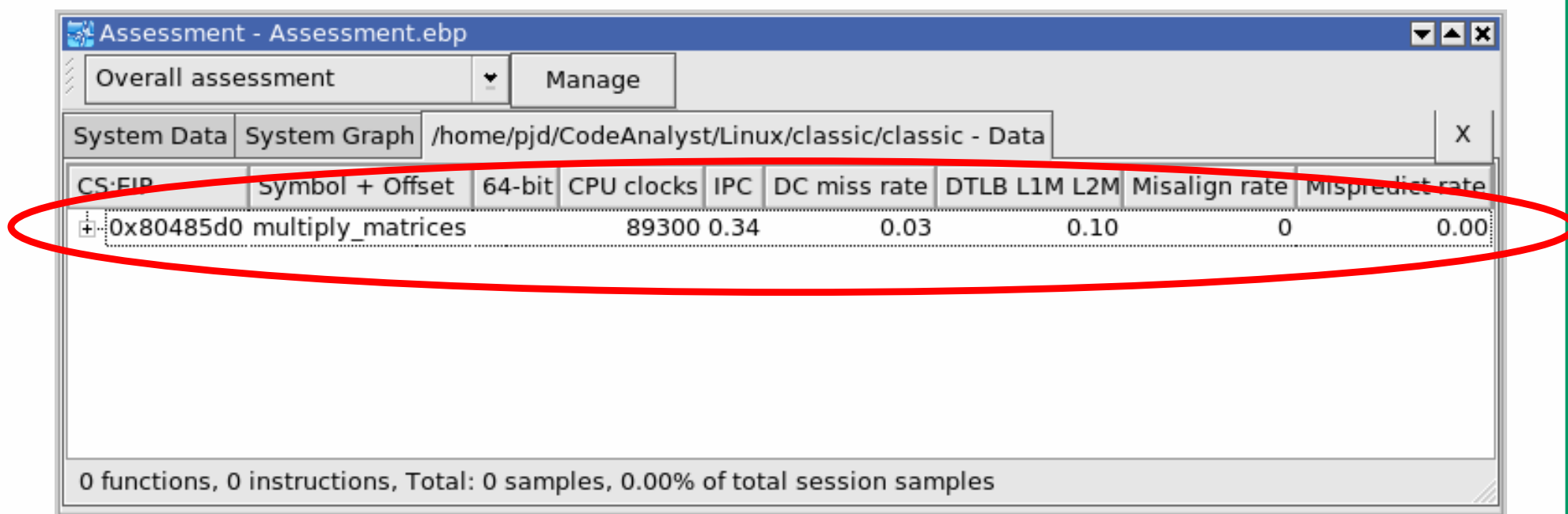
DTLB assessment Manage

System Data System Graph

| Module Name | Ret inst | DC accesses | DTLB L1M L2M | DTLB request rate | DTLB L1M L2M rate | DTLB L1M L2M |
|---|----------|-------------|--------------|-------------------|-------------------|--------------|
| /home/pjd/CodeAnalyst/Linux/classic/classic | 30474 | 7784 | 30259 | 0.26 | 0.10 | 0.39 |
| /no-vmlinux | 400 | 147 | 19 | 0.37 | 0.00 | 0.01 |
| /usr/local/bin/oprofiled | 204 | 40 | 0 | 0.20 | 0 | 0 |
| /lib/libc-2.5.so | 23 | 7 | 1 | 0.30 | 0.00 | 0.01 |
| /usr/lib/xorg/modules/libfb.so | 9 | 2 | 0 | 0.22 | 0 | 0 |
| /usr/lib/qt-3.3/lib/libqt-mt.so.3.3.7 | 9 | 4 | 0 | 0.44 | 0 | 0 |
| /lib/libglib-2.0.so.0.1200.9 | 8 | 8 | 0 | 1 | 0 | 0 |
| /usr/bin/Xorg | 5 | 6 | 0 | 1.20 | 0 | 0 |
| /usr/lib/xorg/modules/libshadow.so | 4 | 2 | 0 | 0.50 | 0 | 0 |
| /usr/lib/libX11.so.6.2.0 | 4 | 3 | 0 | 0.75 | 0 | 0 |
| /lib/libpthread-2.5.so | 4 | 2 | 0 | 0.50 | 0 | 0 |

Event-based profiling

- Drill down to functions within a module
- Find the function responsible for high DTLB miss rate



Assessment - Assessment.ebp

Overall assessment Manage

System Data System Graph /home/pjd/CodeAnalyst/Linux/classic/classic - Data X

| CS:EIP | Symbol + Offset | 64-bit | CPU clocks | IPC | DC miss rate | DTLB L1M L2M | Misalign rate | Mispredict rate |
|-------------|-------------------|--------|------------|------|--------------|--------------|---------------|-----------------|
| + 0x80485d0 | multiply_matrices | | 89300 | 0.34 | 0.03 | 0.10 | 0 | 0.00 |

0 functions, 0 instructions, Total: 0 samples, 0.00% of total session samples

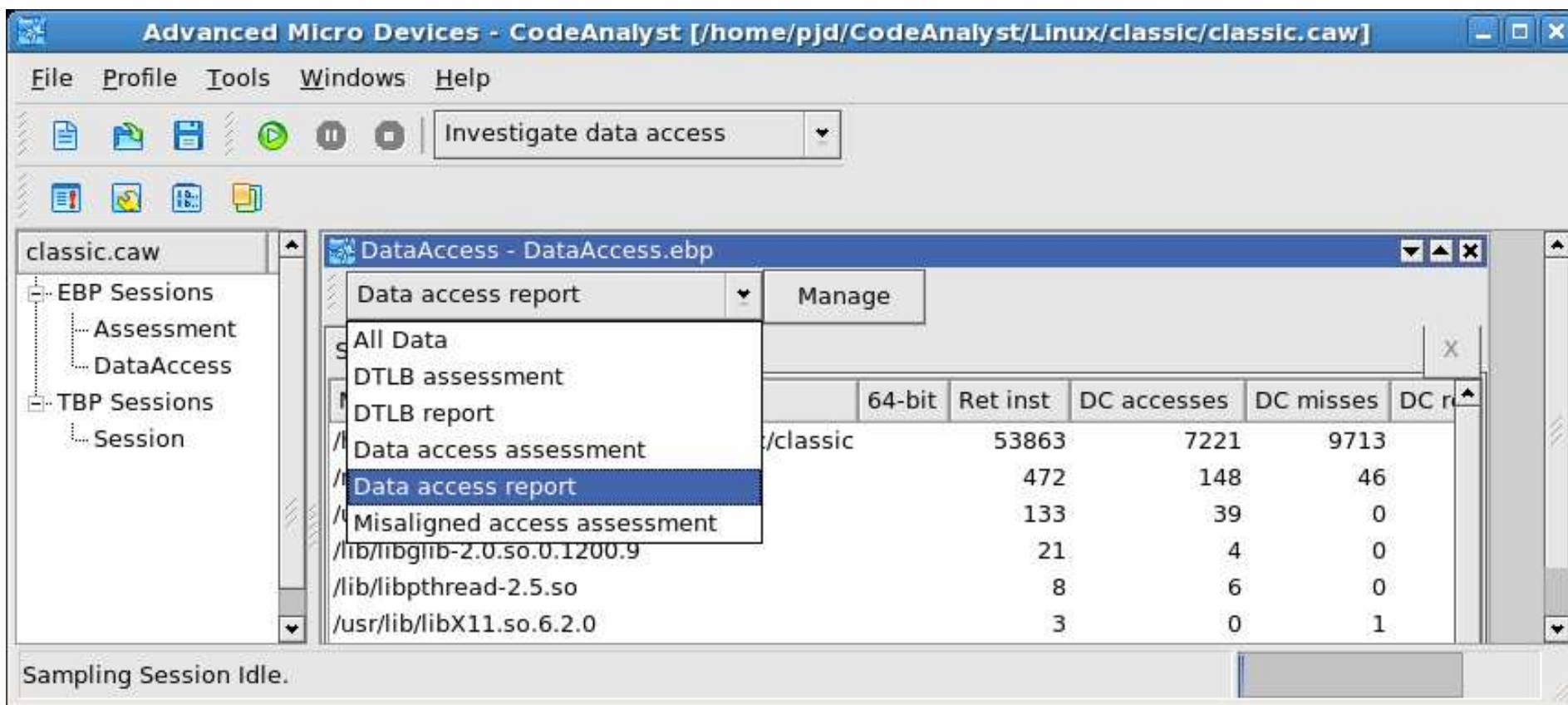
Event-based profiling

- Drill down to the culprit code region

| Line | Source | Code | CPU clocks | IPC | DC miss rate | DTLB L1M L2M | Misalign |
|--|---------------------------------------|-------|------------|------|--------------|--------------|----------|
| 58 | void multiply_matrices() | | 0 | 0 | 0 | 0 | |
| 59 | { | | | | | | |
| 60 | // Multiply the two matrices | | | | | | |
| 61 | for (int i = 0 ; i < ROWS ; i++) { | | | | | | |
| 62 | for (int j = 0 ; j < COLUMNS ; j++) { | | 3 | 1 | 0 | | 0.07 |
| 63 | float sum = 0.0 ; | | | | | | |
| 64 | for (int k = 0 ; k < COLUMNS ; k++) { | | 55628 | 0.33 | 0.03 | | 0.10 |
| 65 | sum = sum + matrix_a[i][k] * m... | | 33074 | 0.35 | 0.03 | | 0.09 |
| | flds (%eax) | d9 00 | 12304 | 0.40 | 0.03 | | 0.09 |
| | fmulb (%ecx) | d8 09 | 10262 | 0.31 | 0.04 | | 0.11 |
| | faddp %st,%st(1) | de c1 | 10508 | 0.34 | 0.03 | | 0.09 |
| 66 | } | | | | | | |
| 67 | matrix_r[i][j] = sum ; | | 584 | 0.33 | 0.03 | | 0.09 |
| 68 | } | | | | | | |
| 0 Instruction, 0 samples, 0.00% of total | | | | | | | |

Event-based profiling

- Use focused profile configurations to investigate specific issues in more depth
- “Data access” concentrates on DC/DTLB behavior



The screenshot shows the AMD CodeAnalyst Linux interface. The main window is titled "Advanced Micro Devices - CodeAnalyst [/home/pjd/CodeAnalyst/Linux/classic/classic.caw]". The menu bar includes File, Profile, Tools, Windows, and Help. The toolbar contains icons for file operations and a dropdown menu set to "Investigate data access".

The left sidebar shows a tree view with "classic.caw" expanded, containing "EBP Sessions" (Assessment, DataAccess) and "TBP Sessions" (Session).

The main pane displays the "DataAccess - DataAccess.ebp" window. A dropdown menu is open, showing options: "All Data", "DTLB assessment", "DTLB report", "Data access assessment", "Data access report" (highlighted), and "Misaligned access assessment". A "Manage" button is visible next to the dropdown.

The "Data access report" table shows the following data:

| | 64-bit | Ret inst | DC accesses | DC misses | DC r |
|------------------------------|--------|----------|-------------|-----------|------|
| /classic | | 53863 | 7221 | 9713 | |
| /lib/libglib-2.0.so.0.1200.9 | | 472 | 148 | 46 | |
| /lib/libpthread-2.5.so | | 133 | 39 | 0 | |
| /usr/lib/libX11.so.6.2.0 | | 21 | 4 | 0 | |
| | | 8 | 6 | 0 | |
| | | 3 | 0 | 1 | |

The status bar at the bottom indicates "Sampling Session Idle."

Event-based profiling

- Example: Data access report
- Shows data cache miss rate and miss ratio

DataAccess - DataAccess.ebp

Data access report Manage

System Data System Graph

| Module Name | Ret inst | DC access | DC miss | DC refills | DC access rate | DC miss rate | DC miss ratio | DC refill rate |
|---|----------|-----------|---------|------------|----------------|--------------|---------------|----------------|
| /home/pjd/CodeAnalyst/Linux/classic/classic | 53863 | 7221 | 9713 | 9712 | 0.27 | 0.04 | 0.13 | 0.04 |
| /no-vmLinux | 472 | 148 | 46 | 45 | 0.63 | 0.02 | 0.03 | 0.02 |
| /usr/local/bin/oprofiled | 133 | 39 | 0 | 0 | 0.59 | 0 | 0 | 0 |
| /lib/libglib-2.0.so.0.1200.9 | 21 | 4 | 0 | 0 | 0.38 | 0 | 0 | 0 |
| /lib/libc-2.5.so | 21 | 5 | 0 | 0 | 0.48 | 0 | 0 | 0 |
| /usr/lib/xorg/modules/libfb.so | 16 | 5 | 0 | 0 | 0.62 | 0 | 0 | 0 |
| /usr/bin/Xorg | 11 | 6 | 0 | 0 | 1.09 | 0 | 0 | 0 |
| /usr/lib/qt-3.3/lib/libqt-mt.so.3.3.7 | 10 | 2 | 2 | 0 | 0.40 | 0.04 | 0.10 | 0 |
| /usr/lib/xorg/modules/libshadow.so | 8 | 8 | 0 | 0 | 2 | 0 | 0 | 0 |

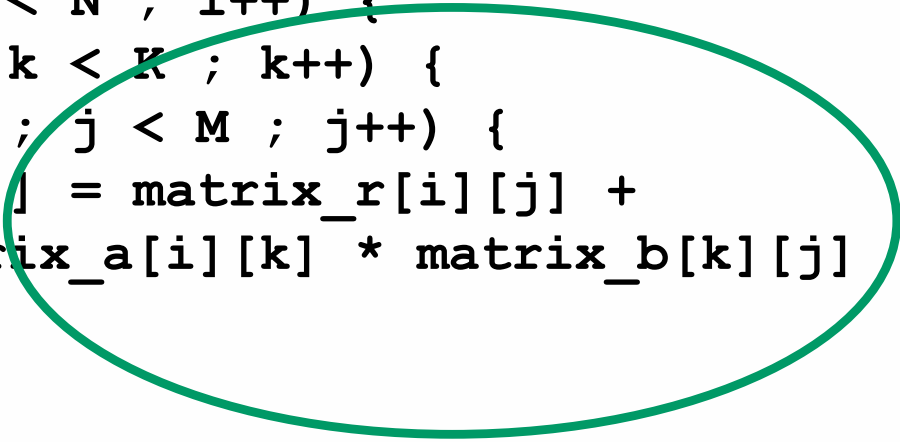
Tuning

Make changes to tune the program

- Change the loop nest
- Use sequential unit strides to avoid DTLB/DC misses

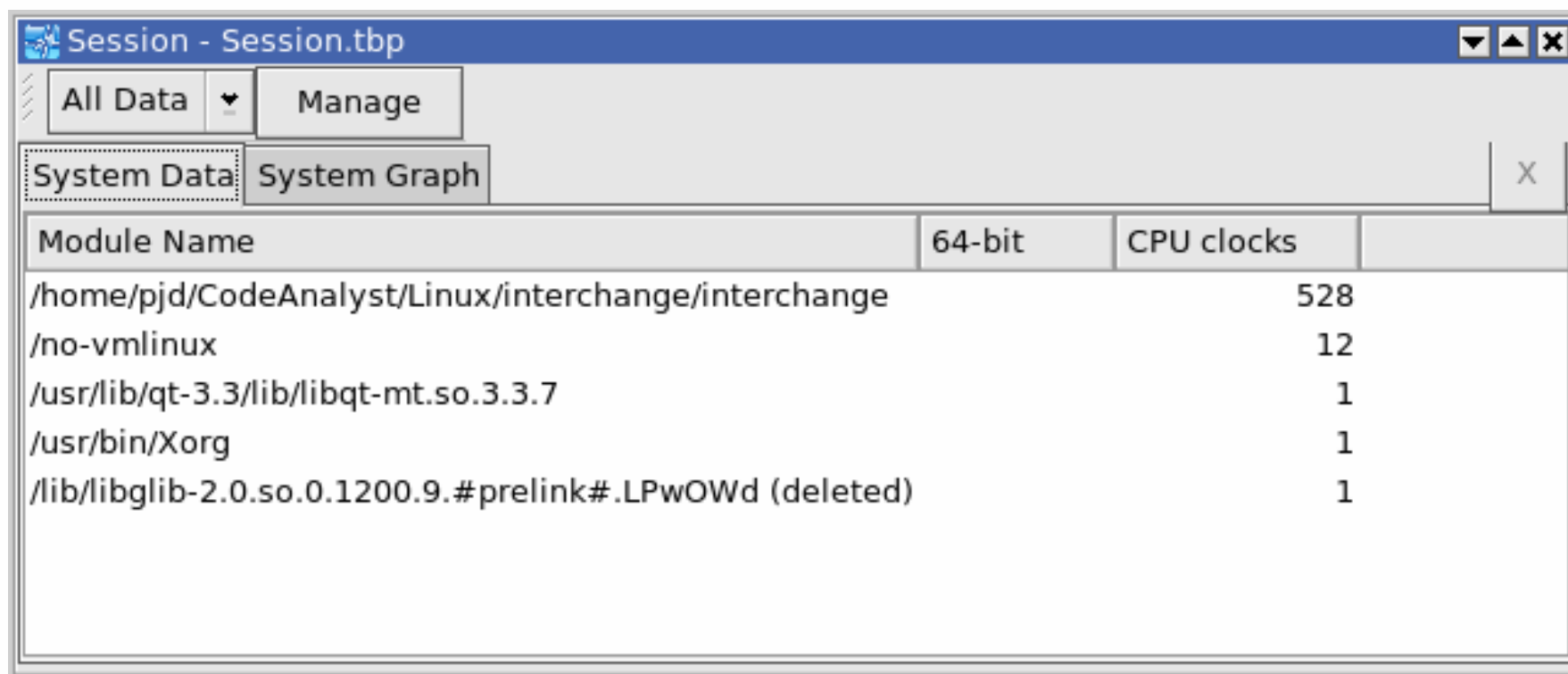
```

void multiply_matrices()
{
    // Multiply the two matrices
    for (int i = 0 ; i < N ; i++) {
        for (int k = 0 ; k < K ; k++) {
            for (int j = 0 ; j < M ; j++) {
                matrix_r[i][j] = matrix_r[i][j] +
                    matrix_a[i][k] * matrix_b[k][j] ;
            }
        }
    }
}
  
```



Measure again and evaluate results

- Execution time is reduced: 10.97 sec → 2.07 sec
- TBP samples are reduced: 8641 → 528



The screenshot shows a window titled 'Session - Session.tbp' with a menu bar containing 'All Data' and 'Manage'. Below the menu bar are two tabs: 'System Data' (selected) and 'System Graph'. The 'System Data' tab displays a table with the following data:

| Module Name | 64-bit | CPU clocks |
|---|--------|------------|
| /home/pjd/CodeAnalyst/Linux/interchange/interchange | | 528 |
| /no-vmLinux | | 12 |
| /usr/lib/qt-3.3/lib/libqt-mt.so.3.3.7 | | 1 |
| /usr/bin/Xorg | | 1 |
| /lib/libglib-2.0.so.0.1200.9.#prelink#.LPwOWd (deleted) | | 1 |

IPC is improved

- DTLB misses are reduced (4 vs. 30259) ...
- ... and IPC is improved (1.94 vs. 0.34)

DTLB assessment

| Module Name | Ret inst | DC accesses | DTLB L1M L2M | DTLB request rate | DTLB L1M L2M rate |
|---|----------|-------------|--------------|-------------------|-------------------|
| /home/pjd/CodeAnalyst/Linux/interchange/interchange | 19299 | 1539 | 4 | 0.08 | 0.08 |
| /no-vmLinux | 130 | 13 | 0 | 0.10 | 0 |
| /usr/local/bin/oprofiled | 68 | 0 | 0 | 0 | 0 |
| /lib/libc-2.5.so | 6 | 0 | 0 | 0 | 0 |
| /usr/lib/libgconf-2.so | | | | | 0 |
| /usr/lib/xorg/modules | | | | | 0 |
| /usr/lib/qt-3.3/plugins | | | | | 0 |

IPC assessment

| Module Name | Ret inst | CPU clocks | IPC | CPI |
|---|----------|------------|------|------|
| /home/pjd/CodeAnalyst/Linux/interchange/interchange | 19299 | 9944 | 1.94 | 0.52 |
| /no-vmLinux | 130 | 346 | 0.38 | 2.66 |
| /usr/local/bin/oprofiled | 68 | 46 | 1.48 | 0.68 |
| /lib/libc-2.5.so | 6 | 7 | 0.86 | 1.17 |
| /usr/lib/libgconf-2.so.4.1.0.#prelink#.MLuDaX (delet... | 4 | 2 | 2 | 0.50 |
| /usr/lib/xorg/modules/libshadow.so | 2 | 8 | 0.25 | 4 |
| /usr/lib/qt-3.3/plugins/styles/bluecurve.so | 2 | 0 | 0 | 0 |
| /usr/lib/libORBit-2.so.0.1.0.#prelink#.sQ6yVu (delet... | 2 | 5 | 0.40 | 2.50 |

How to get AMD CodeAnalyst

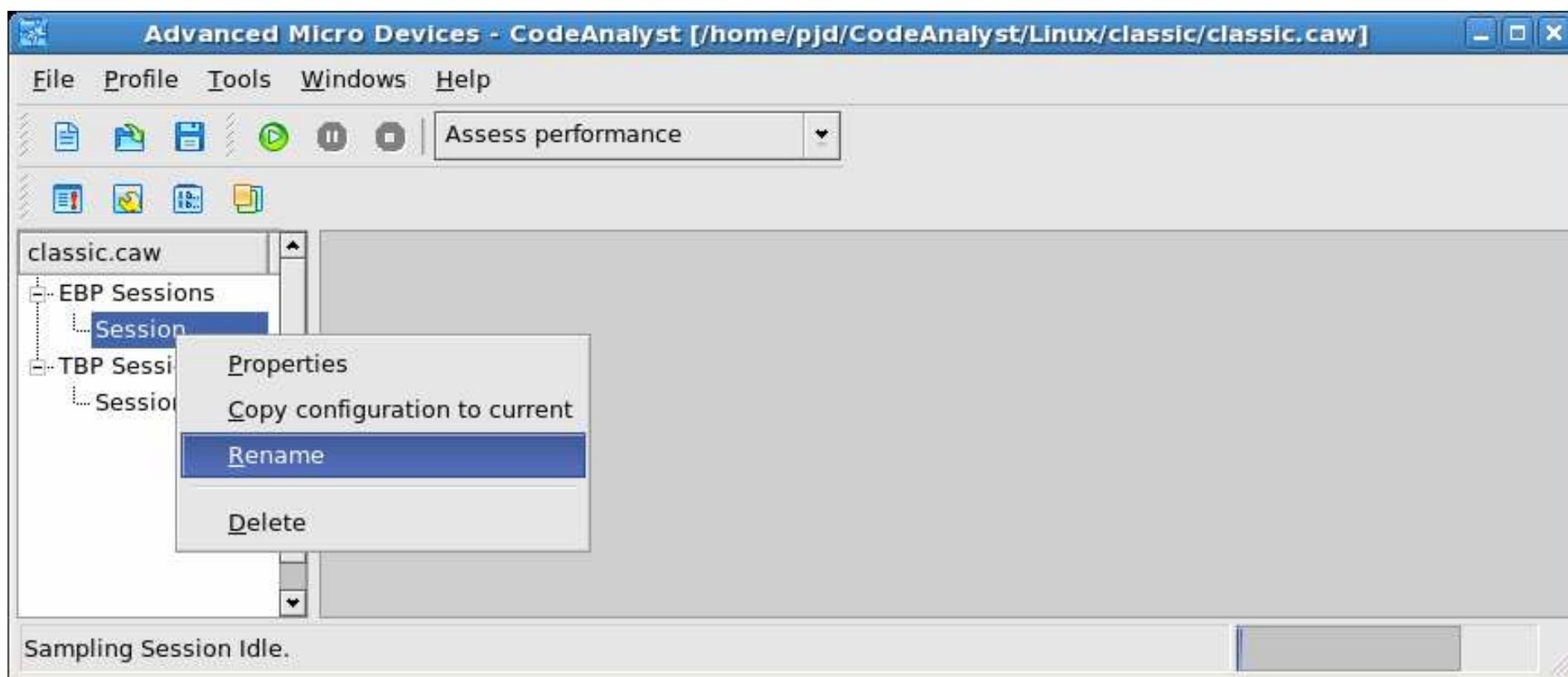
- Free!
- Download from AMD Developer Central
 - <http://developer.amd.com/>
 - <http://www.amd.com/codeanalyst/>
- Available versions
 - V2.5 tarball and RPMs: RHEL 4 (U1, U2, U3)
 - V2.6 tarball and RPMs: RHEL 5 (beta), Fedora Core 5 and 6
- E-mail: support.codeanalyst@amd.com

- Available at AMD Developer Central
- “An introduction to analysis and optimization with AMD CodeAnalyst”
 - <http://developer.amd.com/articles.jsp?id=2&num=1>
- “Basic performance measurements for AMD Athlon™ 64 and AMD Opteron™ Processors”
 - <http://developer.amd.com/articles.jsp?id=90&num=1>
- “BIOS and Kernel Developer’s Guide” or “BKDG” for descriptions of hardware performance events
 - <http://developer.amd.com/documentation.jsp>
- Other AMD tools including performance libraries
 - <http://developer.amd.com/downloads.jsp>

Back-up slides

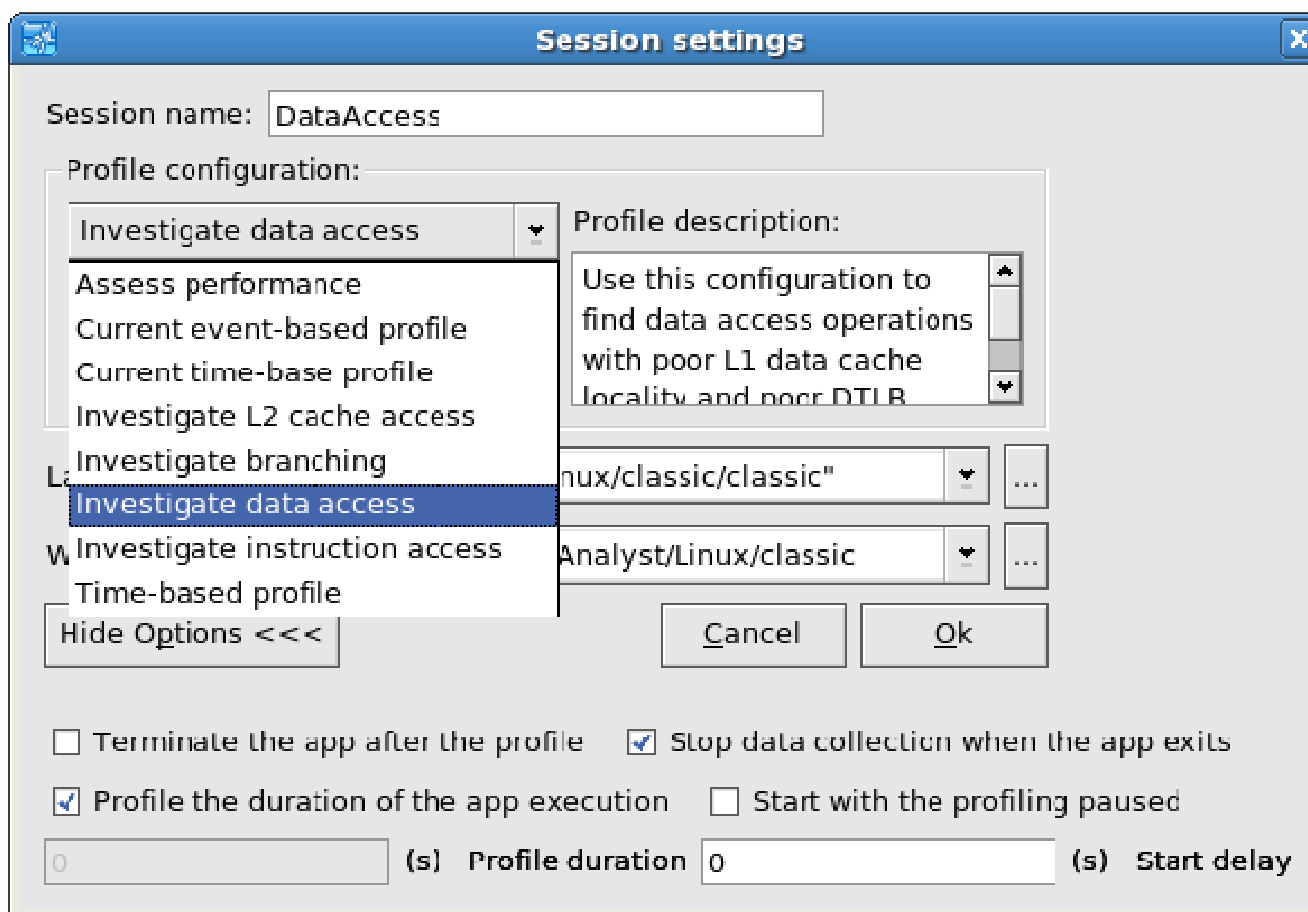
Rename / delete session data

- Right-click on session to name or delete it
- Can also show session properties (such as events)



Change session settings

- Change session name
- Choose a profile (data collection) configuration
- Edit "current" profile configuration to customize



Edit profile configuration

Edit event configuration

Profile name: AMD Athlon(tm) 64 X2 Dual Core Processor 4400+ - Family 15 Model 35 Stepping 2

Event configurations

Configuration description:

Use this configuration to find data access operations with poor L1 data cache locality and poor DTLB behavior.

Individual Events

- [0] Dispatched FPU operations
- [1] Cycles with no FPU ops retired
- [20] Segment register loads
- [21] Pipeline restart due to self-modifying code
- [22] Pipeline restart due to probe hit

Events in configuration

Groups - Events

- [40] Data cache accesses
- [41] Data cache misses
- [42] Data cache refills from L2 or system
- [c0] Retired Instructions
- [45] L1 DTLB miss and L2 DTLB hit
- [46] L1 DTLB miss and L2 DTLB miss
- [47] Misaligned accesses
- [c0] Retired Instructions

Unit Masks

- ☐ Reserved
- ☐ Reserved
- ☐ Reserved
- ☐ Reserved
- ☐ Reserved
- ☐ Reserved
- ☐ Reserved
- ☐ Reserved

Event count: ☒ User ☒ OS ☐ Edge ☐

Multiplexing interval ms

Event-based profiling

- The “All data” view shows all event (sample) counts

The screenshot shows the AMD CodeAnalyst Linux interface. The main window is titled "Advanced Micro Devices - CodeAnalyst [/home/pjd/CodeAnalyst/Linux/classic/classic.caw]". The menu bar includes File, Profile, Tools, Windows, and Help. The toolbar contains icons for file operations, a play button, a pause button, and a stop button, along with a dropdown menu set to "Assess performance".

On the left, a tree view shows the project structure: classic.caw, EBP Sessions, Session, TBP Sessions, and Session. The main pane displays the "Session - Session.ebp" window, which is set to the "All Data" view. Below this, there are tabs for "System Data" and "System Graph". The "System Data" tab is active, showing a table of performance metrics for various modules.

| Module Name | 64-bit | CPU clocks | DC accesses | DC misses | DTL |
|---|--------|------------|-------------|-----------|-----|
| /home/pjd/CodeAnalyst/Linux/classic/classic | | 89300 | 7784 | 10489 | |
| /no-vmlinux | | 2684 | 147 | 28 | |
| /usr/local/bin/oprofiled | | 165 | 40 | 0 | |
| /lib/libc-2.5.so | | 43 | 7 | 2 | |
| /usr/lib/xorg/modules/libshadow.so | | 31 | 2 | 0 | |
| /usr/lib/qt-3.3/lib/libqt-mt.so.3.3.7 | | 30 | 4 | 1 | |
| /lib/libglib-2.0.so.0.1200.9 | | 24 | 8 | 1 | |
| /usr/bin/Xorg | | 21 | 6 | 2 | |
| /lib/libpthread-2.5.so | | 17 | 2 | 1 | |
| /usr/lib/libX11.so.6.2.0 | | 16 | 3 | 0 | |

At the bottom of the window, a status bar indicates "Sampling Session Idle." and a progress bar.

Trademark Attribution

AMD, the AMD Arrow logo, AMD Athlon, AMD Opteron and combinations thereof are trademarks of Advanced Micro Devices, Inc. in the United States and/or other jurisdictions. Linux is a registered trademark of Linus Torvalds. Other names used in this presentation are for identification purposes only and may be trademarks of their respective owners.

©2007 Advanced Micro Devices, Inc. All rights reserved.